Open-Source Software Practice

Lab 04. Git Misc. + Code Editor

Instructor: Jaemin Jo (조재민, jmjo@skku.edu)
Interactive Data Computing Lab (*IDCLab*),
College of Computing and Informatics,
Sungkyunkwan University

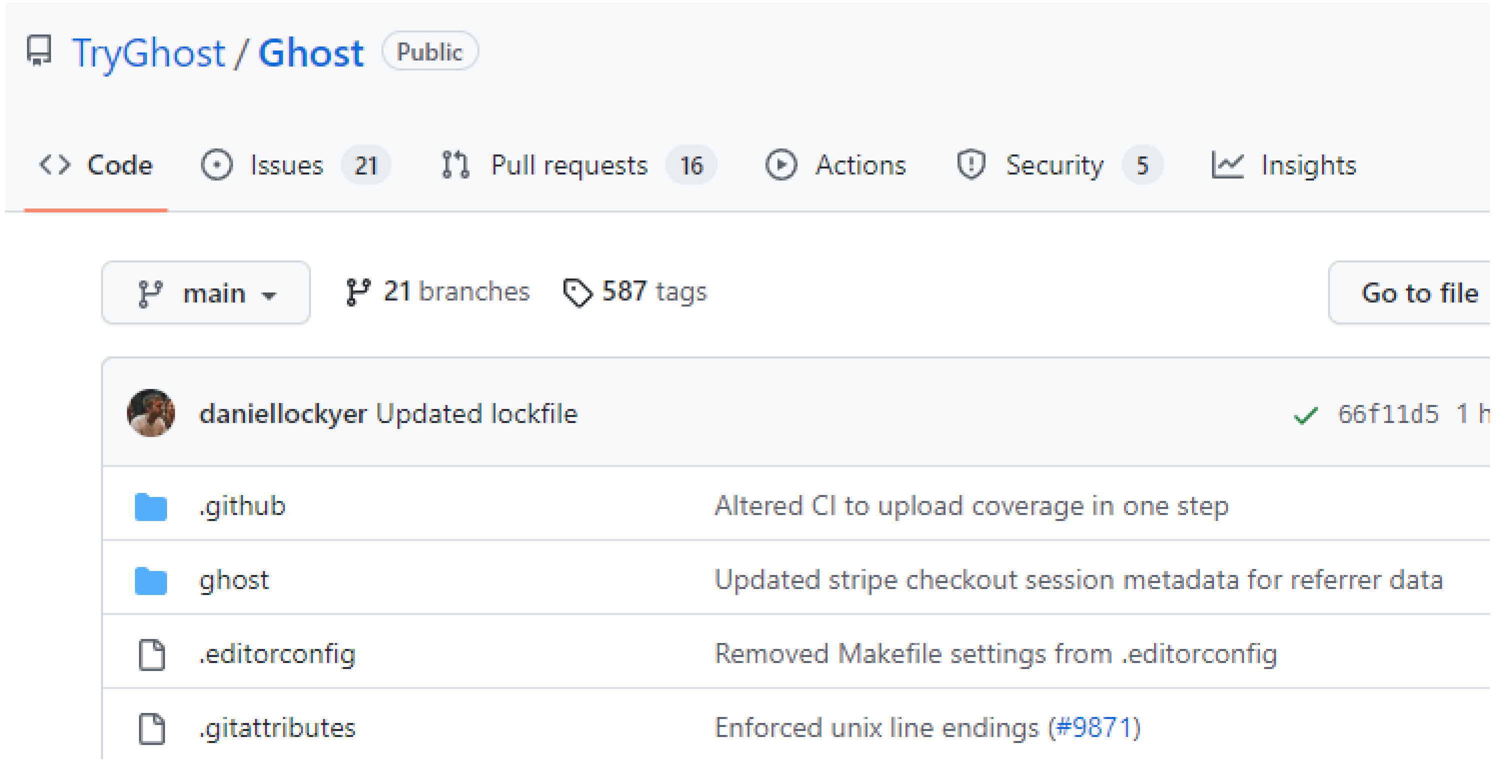
Goals

1. EditorConfig
2. IntelliCode
3. Tagging
4. Diff
5. Stashing
6. Reset
7. Rebase

Setting up

1. Initialize a new Git repo: `git init`
2. echo “Hello, World!” >> README.md
3. `git add .`
4. `git commit –m “Added README.md”`
5. `git remote add origin git@github.com:e–/test.git`
6. `git push ––set–upstream origin main`

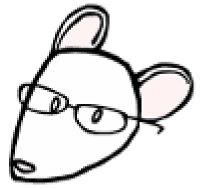
EditorConfig


TryGhost / Ghost Public

daniellockyer Updated lockfile ✓ 66f11d5 1 h

.github	Altered CI to upload coverage in one step
ghost	Updated stripe checkout session metadata for referrer data
.editorconfig	Removed Makefile settings from .editorconfig
.gitattributes	Enforced unix line endings (#9871)

EditorConfig

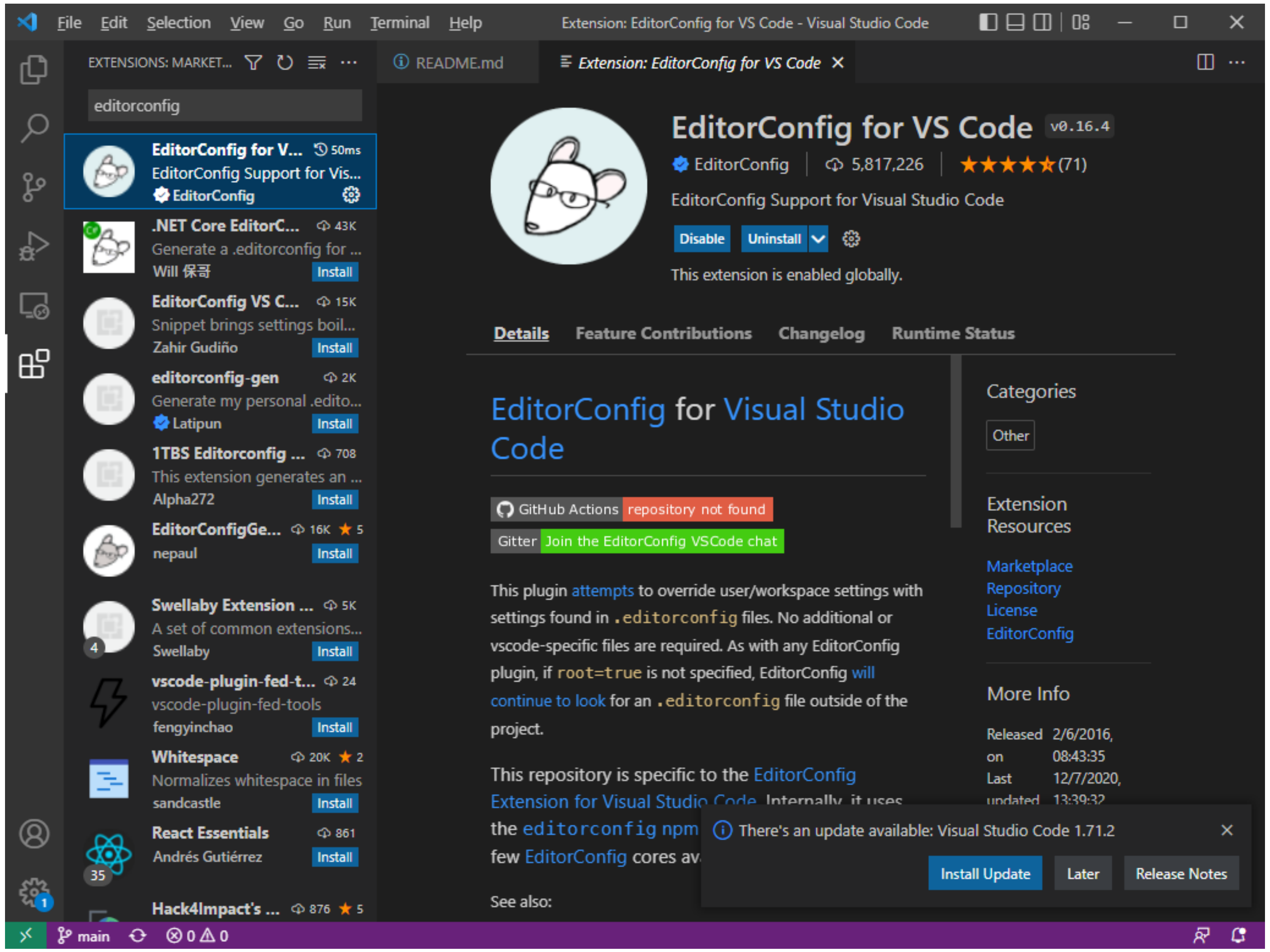


- Why EditorConfig?
  - To maintain consistent coding styles for multiple developers working on the same project across various editors and IDEs.
  - Tab for indent, Tab size: 4, utf-8
  - Space for indent, Tab size: 2, utf-8
  - Space for indent, Tab size: 4, euc-kr

- Specify end of line, character set, indent style, indent size, and tab width used throughout a project.

- Official: https://editorconfig.org/

EditorConfig


File Edit Selection View Go Run Terminal Help
Extension: EditorConfig for VS Code - Visual Studio Code
EXTENSIONS: MARKET...
editorconfig
README.md
Extension: EditorConfig for VS Code
Disable
Uninstall
Install Update
Later
Release Notes
main
0 0

EditorConfig for V...	50ms	EditorConfig Support for Vis... EditorConfig
.NET Core EditorC...	43K	Generate a .editorconfig for ... Will 保哥 Install
EditorConfig VS C...	15K	Snippet brings settings boil... Zahir Gudiño Install
editorconfig-gen	2K	Generate my personal .edito... Latipun Install
1TBS Editorconfig ...	708	This extension generates an ... Alpha272 Install
EditorConfigGe...	16K ★ 5	nepaul Install
Swellaby Extension ...	5K	A set of common extensions... Swellaby Install
vscode-plugin-fed-t...	24	vscode-plugin-fed-tools fengyinchao Install
Whitespace	20K ★ 2	Normalizes whitespace in files sandcastle Install
React Essentials	861	Andrés Gutiérrez Install
Hack4Impact's ...	876 ★ 5	

EditorConfig for VS Code v0.16.4
EditorConfig | 5,817,226 | ★★★★★ (71)
EditorConfig Support for Visual Studio Code
This extension is enabled globally.

Details | Feature Contributions | Changelog | Runtime Status

EditorConfig for Visual Studio Code

GitHub Actions repository not found
Gitter Join the EditorConfig VSCode chat

This plugin attempts to override user/workspace settings with settings found in `.editorconfig` files. No additional or vscode-specific files are required. As with any EditorConfig plugin, if `root=true` is not specified, EditorConfig will continue to look for an `.editorconfig` file outside of the project.

This repository is specific to the EditorConfig Extension for Visual Studio Code. Internally, it uses the `editorconfig` npm ... few EditorConfig cores av...

See also:

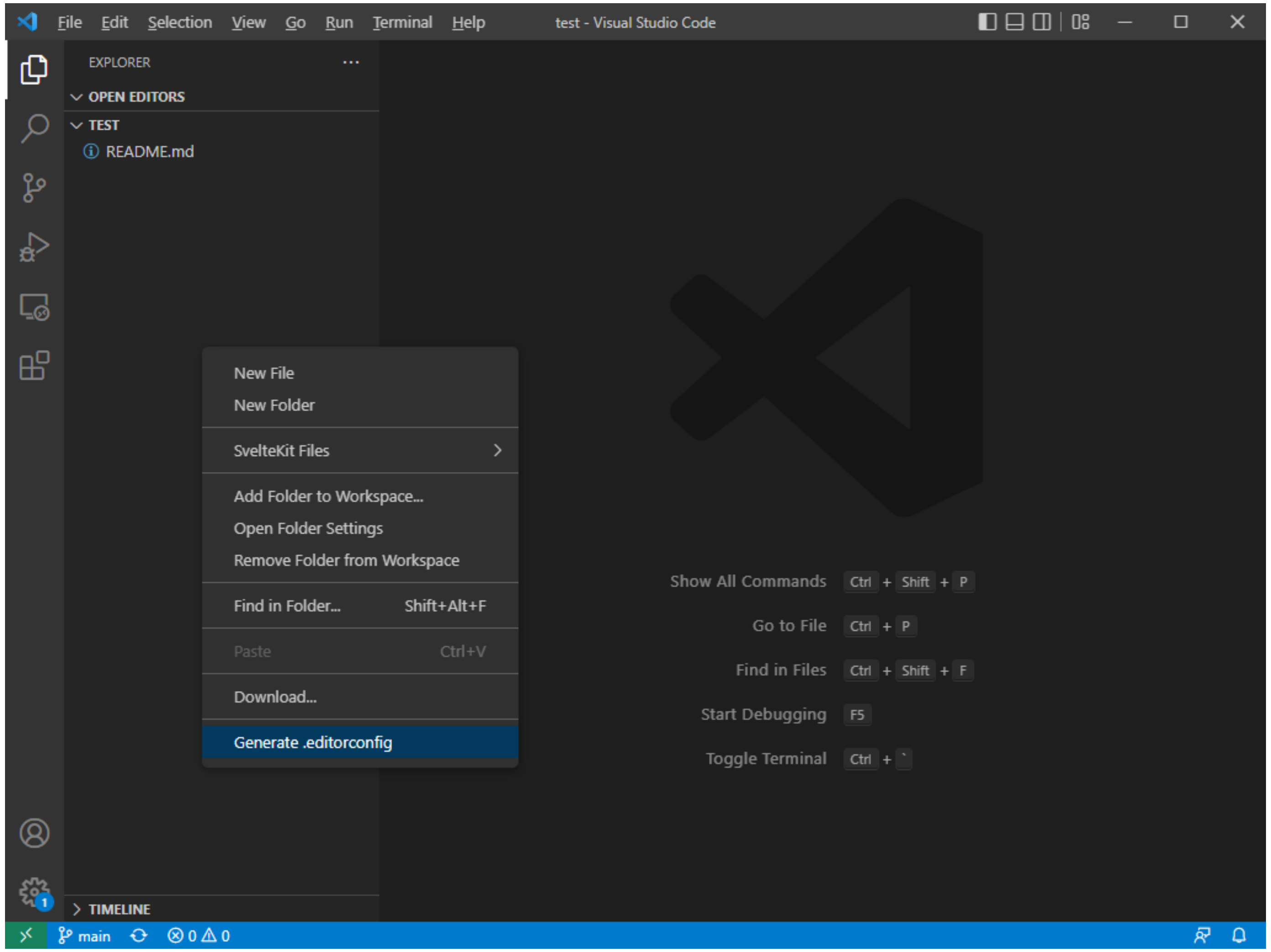
Categories
Other

Extension Resources
Marketplace
Repository
License
EditorConfig

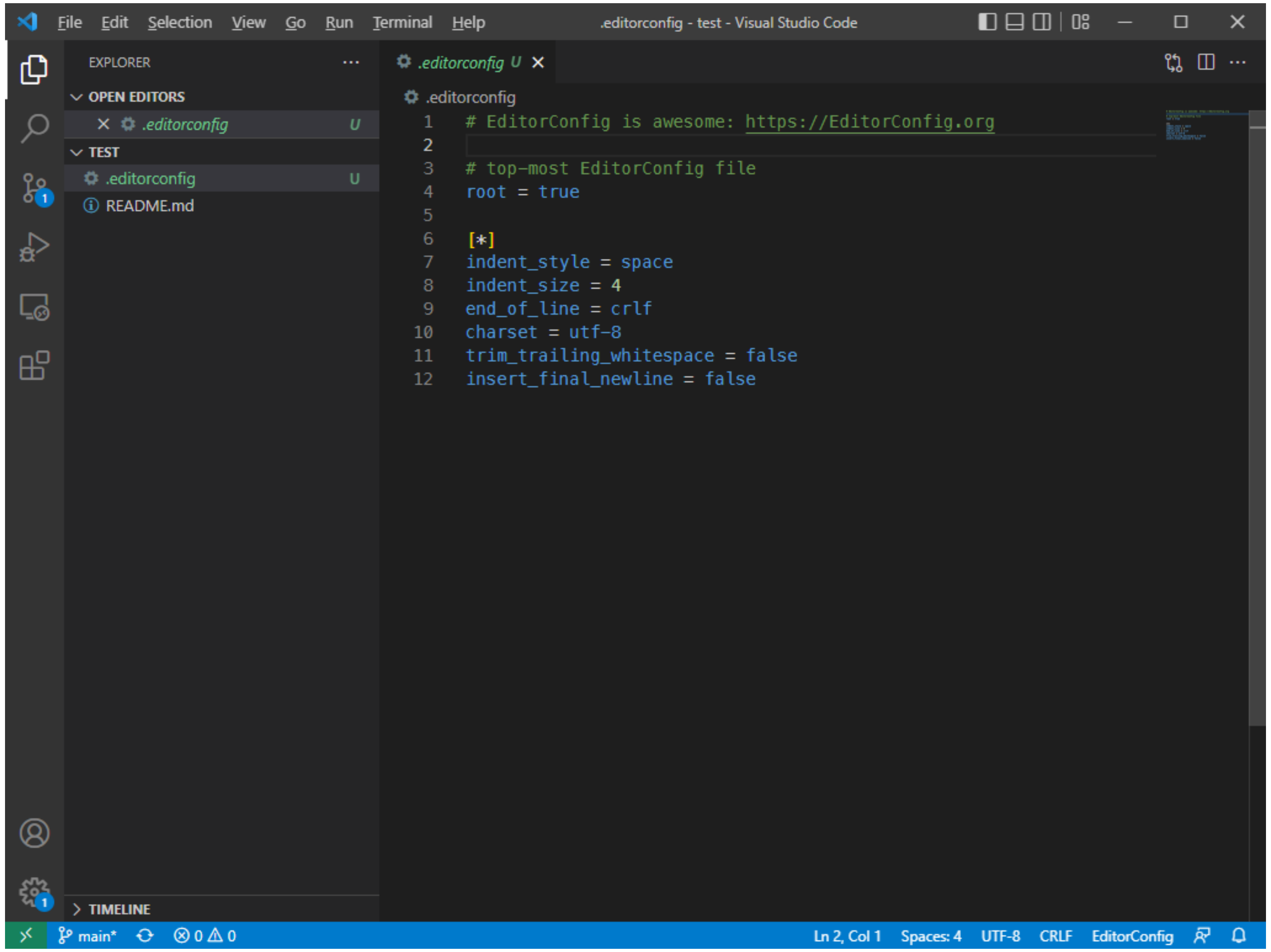
More Info
Released on 2/6/2016, 08:43:35
Last updated 12/7/2020, 13:39:32

There's an update available: Visual Studio Code 1.71.2

EditorConfig



EditorConfig


File Edit Selection View Go Run Terminal Help
.editorconfig - test - Visual Studio Code
EXPLORER
OPEN EDITORS
.editorconfig U
TEST
.editorconfig U
README.md
TIMELINE
main*
Ln 2, Col 1 Spaces: 4 UTF-8 CRLF EditorConfig

```
# EditorConfig is awesome: https://EditorConfig.org

# top-most EditorConfig file
root = true

[*]
indent_style = space
indent_size = 4
end_of_line = crlf
charset = utf-8
trim_trailing_whitespace = false
insert_final_newline = false
```

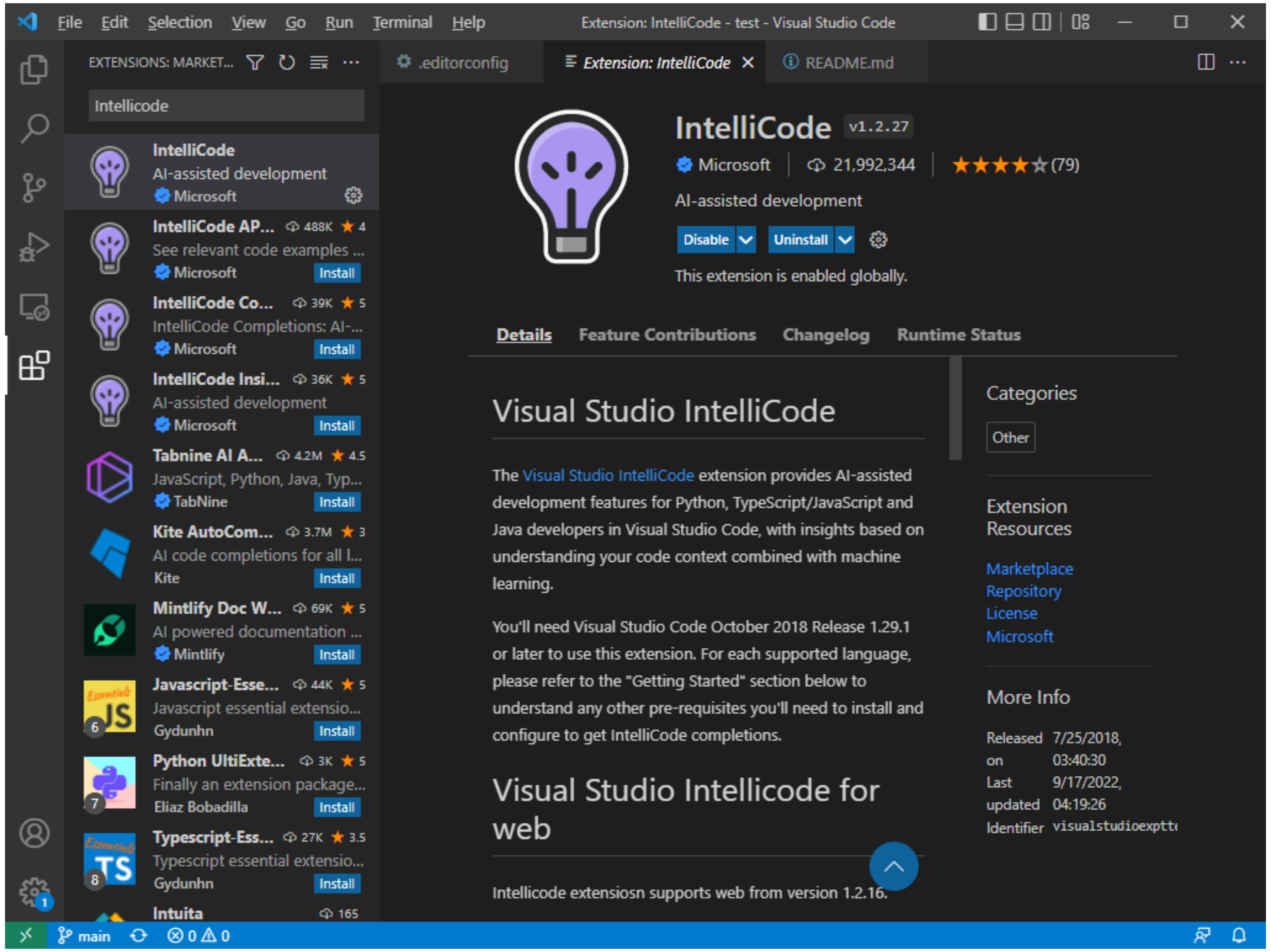
EditorConfig

- Change *indent_size* to 2 and check if it affects the VCS's setting.

- You must add .editorconfig as part of your project.
  - `git add .editorconfig`

- Take a look at .editorconfig used in Airbnb
  - https://github.com/airbnb/javascript/blob/master/.editorconfig

IntelliCode



Intellicode

IntelliCode
AI-assisted development
Microsoft

IntelliCode AP... 488K ★ 4
See relevant code examples ...
Microsoft Install

IntelliCode Co... 39K ★ 5
IntelliCode Completions: AI-...
Microsoft Install

IntelliCode Insi... 36K ★ 5
AI-assisted development
Microsoft Install

Tabnine AI A... 4.2M ★ 4.5
JavaScript, Python, Java, Typ...
TabNine Install

Kite AutoCom... 3.7M ★ 3
AI code completions for all l...
Kite Install

Mintlify Doc W... 69K ★ 5
AI powered documentation ...
Mintlify Install

Javascript-Esse... 44K ★ 5
Javascript essential extensio...
Gydunhn Install

Python UltiExte... 3K ★ 5
Finally an extension package...
Eliaz Bobadilla Install

Typescript-Ess... 27K ★ 3.5
Typescript essential extensio...
Gydunhn Install

Intuita 165

IntelliCode v1.2.27
Microsoft | 21,992,344 | ★★★★☆ (79)
AI-assisted development
Disable | Uninstall
This extension is enabled globally.

Visual Studio IntelliCode

The Visual Studio IntelliCode extension provides AI-assisted development features for Python, TypeScript/JavaScript and Java developers in Visual Studio Code, with insights based on understanding your code context combined with machine learning.

You'll need Visual Studio Code October 2018 Release 1.29.1 or later to use this extension. For each supported language, please refer to the "Getting Started" section below to understand any other pre-requisites you'll need to install and configure to get IntelliCode completions.

Visual Studio Intellicode for web

Intellicode extensiosn supports web from version 1.2.16.

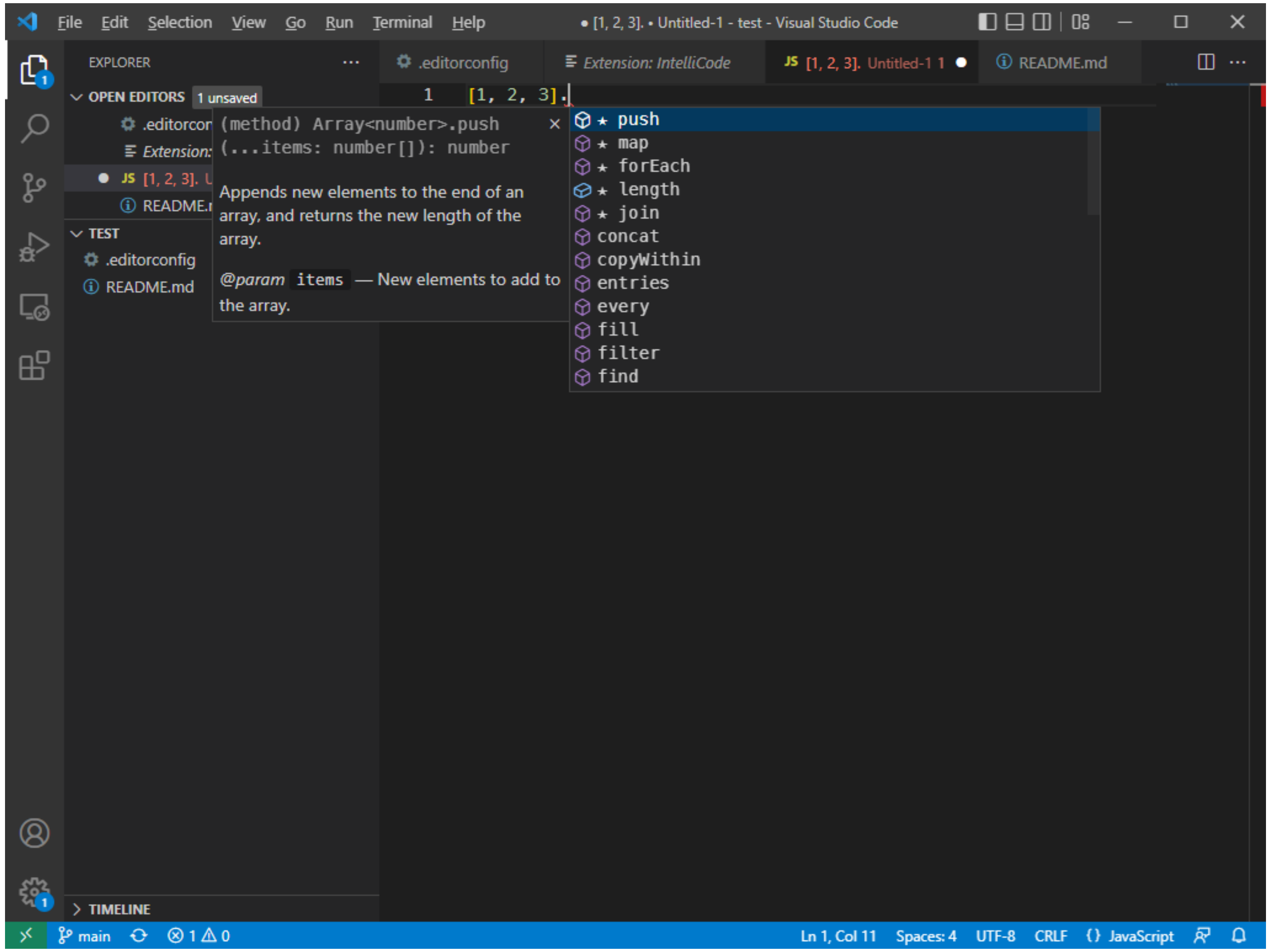
Categories
Other

Extension Resources
Marketplace
Repository
License
Microsoft

More Info

Released on	7/25/2018, 03:40:30
Last updated	9/17/2022, 04:19:26
Identifier	visualstudioexptt

IntelliCode



```
[1, 2, 3].
```

```
(method) Array<number>.push
(...items: number[]): number
```

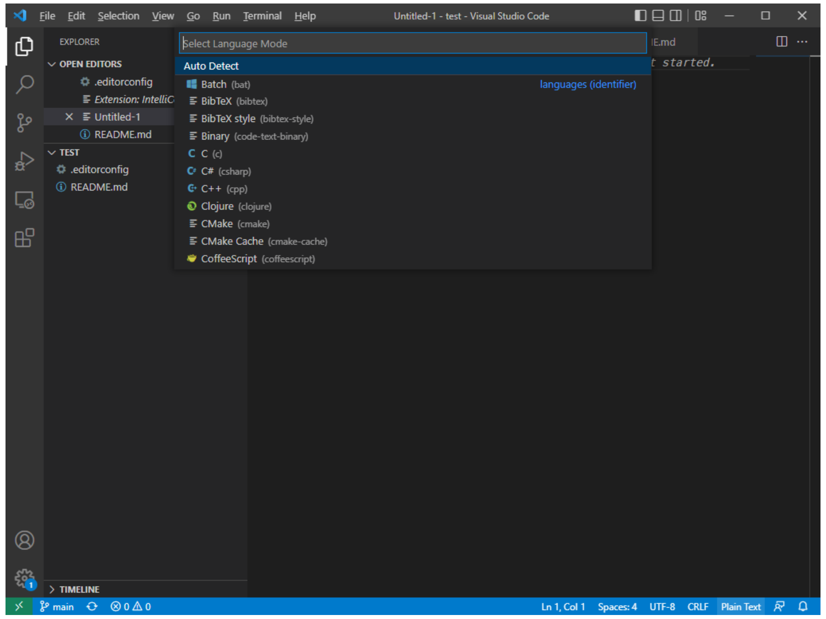
Appends new elements to the end of an array, and returns the new length of the array.

@param `items` — New elements to add to the array.

IntelliCode

- **Q:** IntelliCode does not work!
- **A:** It should detect the language you are using.

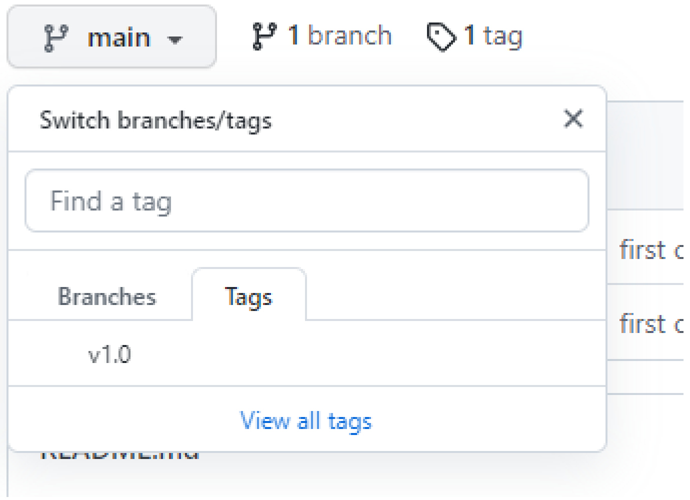
- Let it guess from the file extension (*main.**js***) or
- Select the language mode (Ctrl + K then M)



Let’s Practice – 1 (tagging)

- Set the remote repository first!

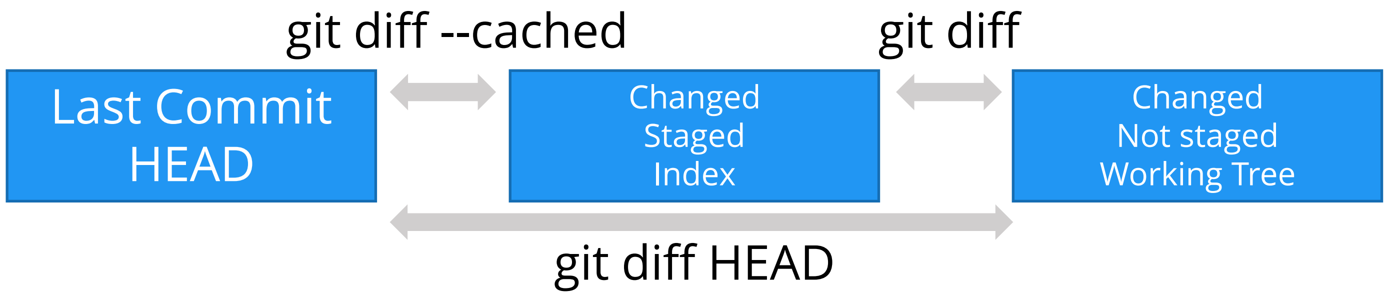
- `git tag –a v0.1 –m “first release”`
- `git push origin v0.1`



Let’s Practice – 2 (diff)

1. Make sure everything has been committed.
2. Append “staging” to README.md: `echo “staging” >> README.md`
3. `git add README.md` (don’t make a commit for now)
4. Open README.md and change “staging” to “working tree”.
5. `git diff`
6. `git diff --cached`
7. `git diff HEAD`

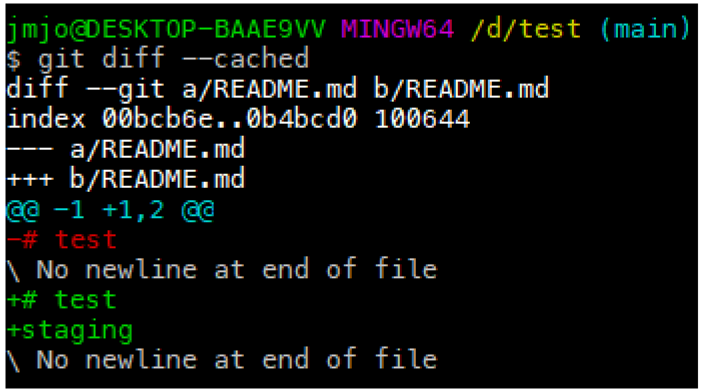
Let's Practice – 2 (diff)



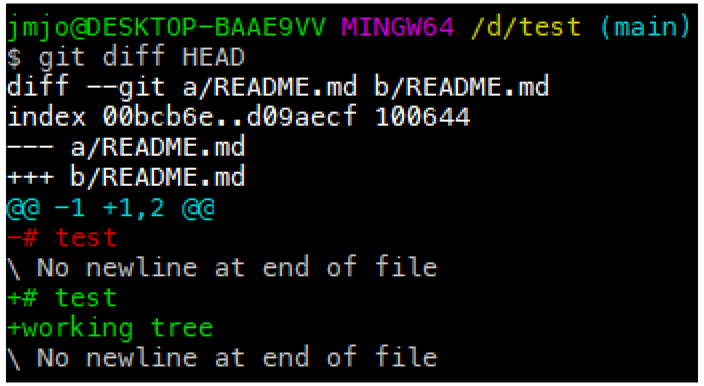
test

test
staging

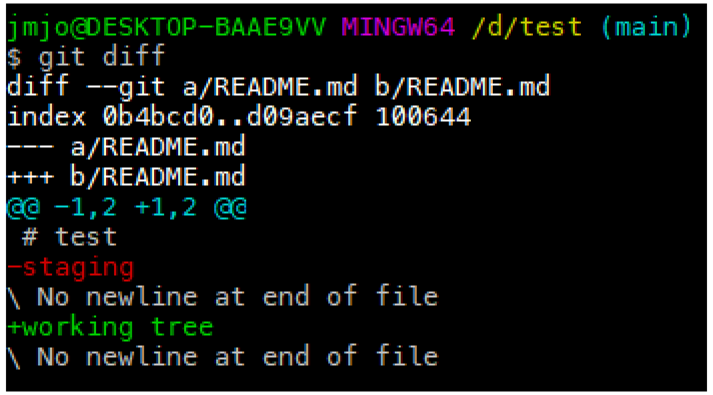
test
working tree



```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (main)
$ git diff --cached
diff --git a/README.md b/README.md
index 00bcb6e..0b4bcd0 100644
--- a/README.md
+++ b/README.md
@@ -1 +1,2 @@
-# test
\ No newline at end of file
+# test
+staging
\ No newline at end of file
```



```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (main)
$ git diff HEAD
diff --git a/README.md b/README.md
index 00bcb6e..d09aecf 100644
--- a/README.md
+++ b/README.md
@@ -1 +1,2 @@
-# test
\ No newline at end of file
+# test
+working tree
\ No newline at end of file
```



```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (main)
$ git diff
diff --git a/README.md b/README.md
index 0b4bcd0..d09aecf 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,2 @@
 # test
-staging
\ No newline at end of file
+working tree
\ No newline at end of file
```

Let’s Practice – 3 (stashing)

- Make sure everything has been committed.

1. Create *main.js* and enter the following code.
2. Commit *main.js*
3. Suppose you are writing the *div* function. Enter “let x = “ in the *div* function.
   - The code is obviously incomplete.

```
function div(a, b) {

}

console.log(div(10, 5));
```

Commit this code

```
function div(a, b) {
    let x =
}

console.log(div(10, 5));
```

Edit it like this

Let’s Practice – 3 (stashing)

- Suppose someone comes to you and asks you to write another function *add*.

4. `git stash`
5. Check if the uncommitted changes are reverted.
6. Write the *add* function as below.
7. Add and commit

```
function div(a, b) {

}

function add(a, b) {
    return a + b;
}

console.log(div(10, 5));
```

Let’s Practice – 3 (stashing)

8. `git stash pop`
9. Finish the *div* function and commit.

```
function div(a, b) {
    let x =
}

function add(a, b) {
    return a + b;
}

console.log(div(10, 5));
```

Code after git stash pop

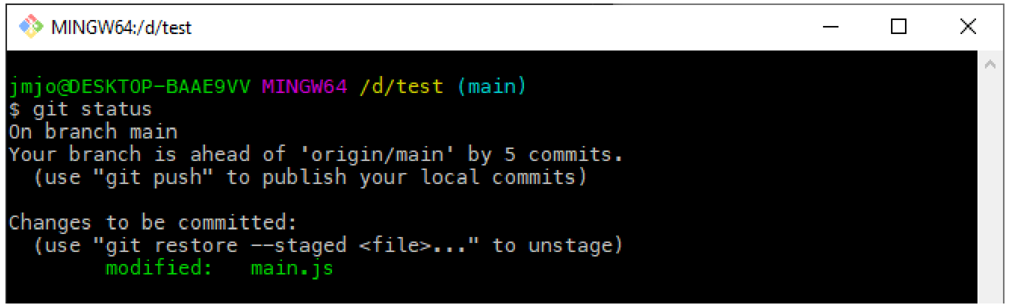
```
function div(a, b) {
    return a / b;
}

function add(a, b) {
    return a + b;
}

console.log(div(10, 5));
```

The final code

Let’s Practice – 4 (soft reset)

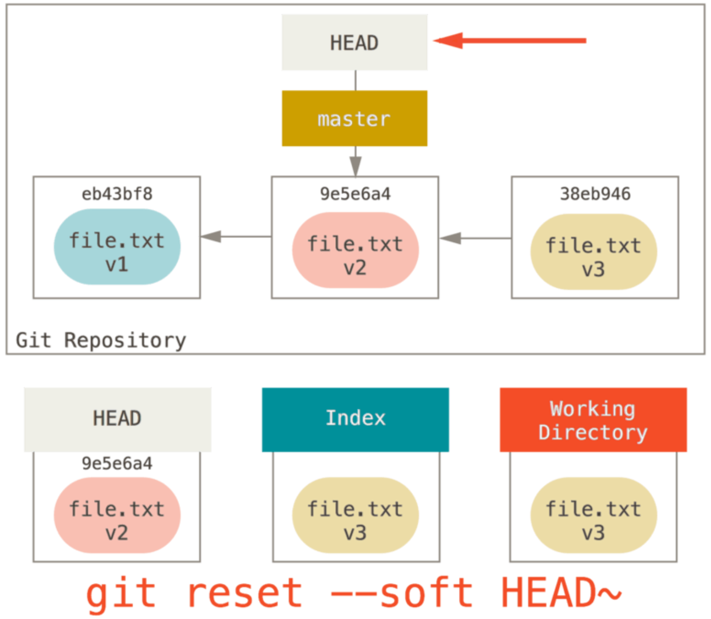
- Make sure everything has been committed.

1. Delete all lines in *main.js* and commit. You just made a big mistake.
2. `git reset --soft HEAD~`
3. `git status`



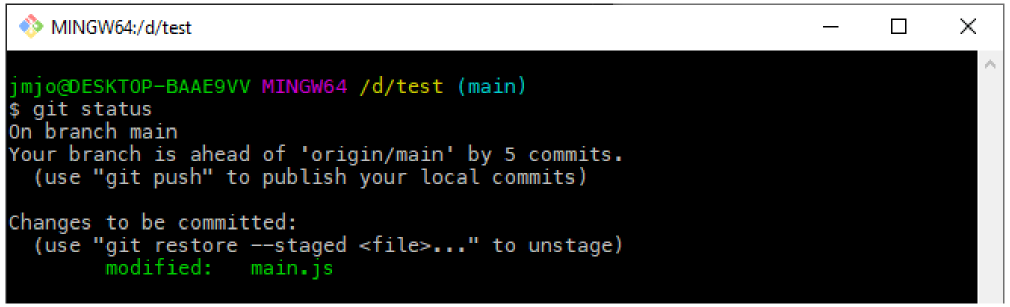
MINGW64:/d/test

```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.js
```



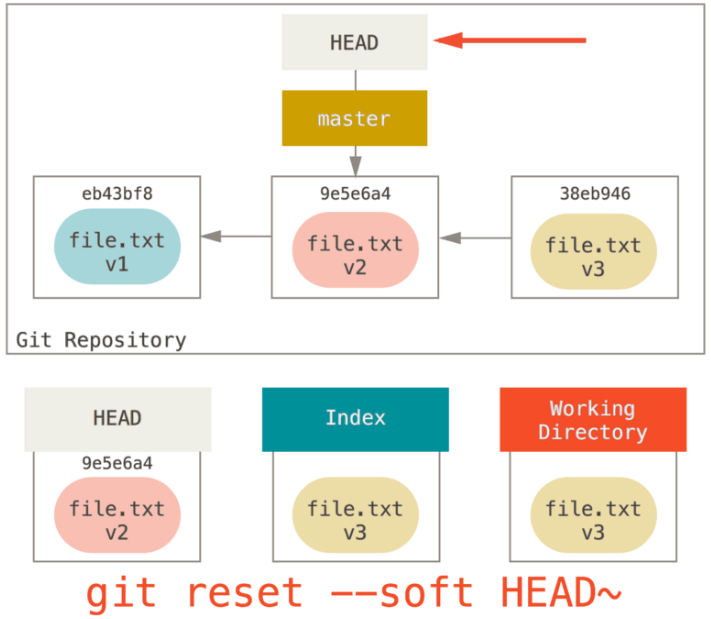
Let’s Practice – 4 (soft reset)

1. Don’t commit. If you do that, you are making the same mistake again.
2. `git restore --staged main.js`
3. `git restore main.js`



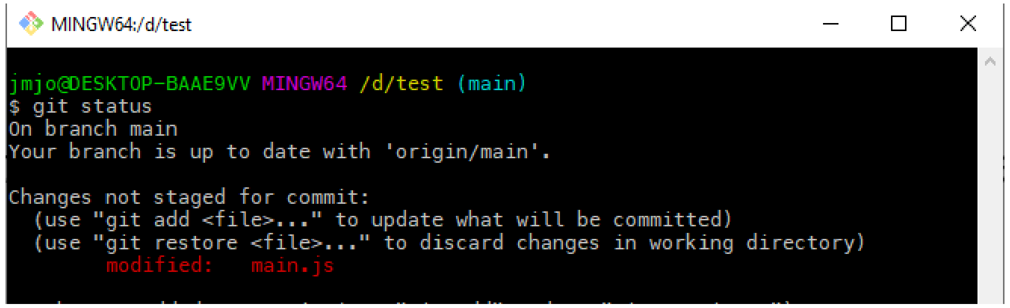
MINGW64:/d/test

```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.js
```

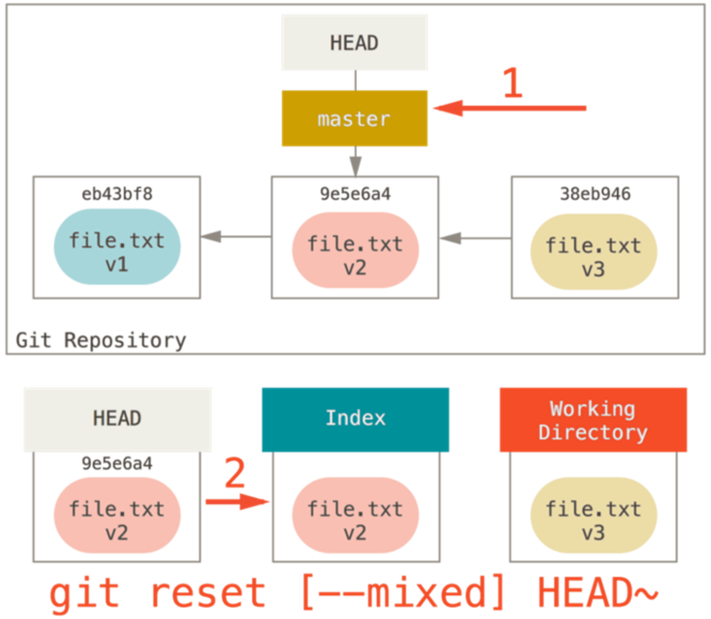


Let’s Practice – 5 (mixed reset)

1. Again, delete all lines in *main.js* and commit. You just made a big mistake.
2. `git reset HEAD~`
3. `git status`
4. `git restore main.js`

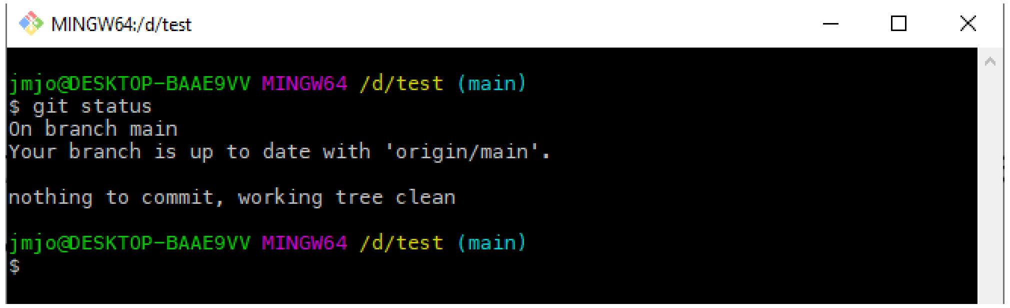


```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   main.js
```

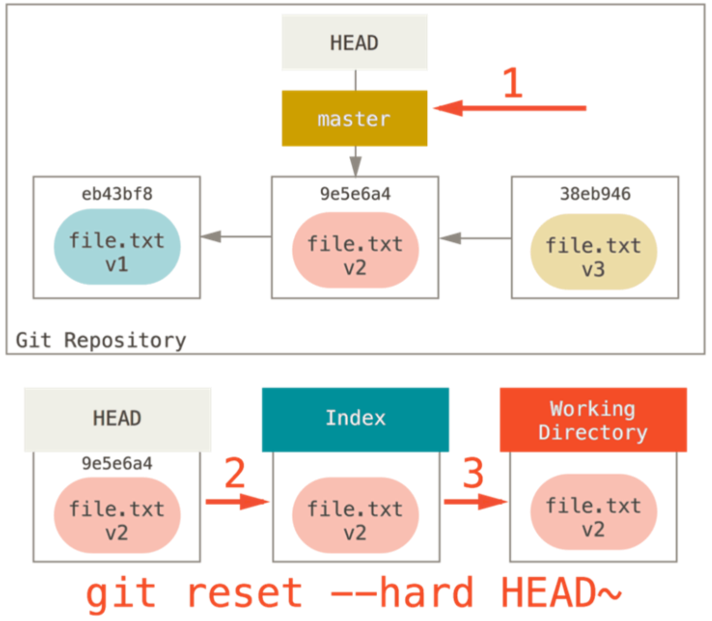


Let's Practice – 6 (hard reset)

1. Again, delete all lines in *main.js* and commit. You just made a big mistake.
2. `git reset --hard HEAD~`
3. `git status`
4. Make sure that *README.md* has come back.



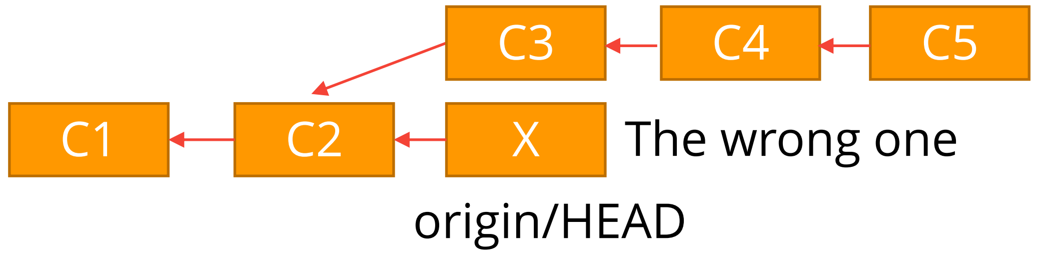
```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (main)
$
```



Let’s Compare

- If you did something wrong and committed the changes:
  - git reset --soft HEAD~ (rarely used)
  - git reset HEAD~ (“Give me a second chance. I will modify and commit it again”)
  - git reset --hard HEAD~ (“I was totally wrong. Reset everything to the second last commit”)

- Did something wrong && committed && pushed:
  - You can reset, but after making commits from the reset HEAD, you need to *git pull* to merge origin/HEAD and HEAD (diverged)



Merge between X and C5 is needed!

Let’s Practice – 7 (rebase)

- Your *main.js* should look like this:

```
function div(a, b) {
    return a / b;
}

function add(a, b) {
    return a + b;
}

console.log(div(10, 5));
```

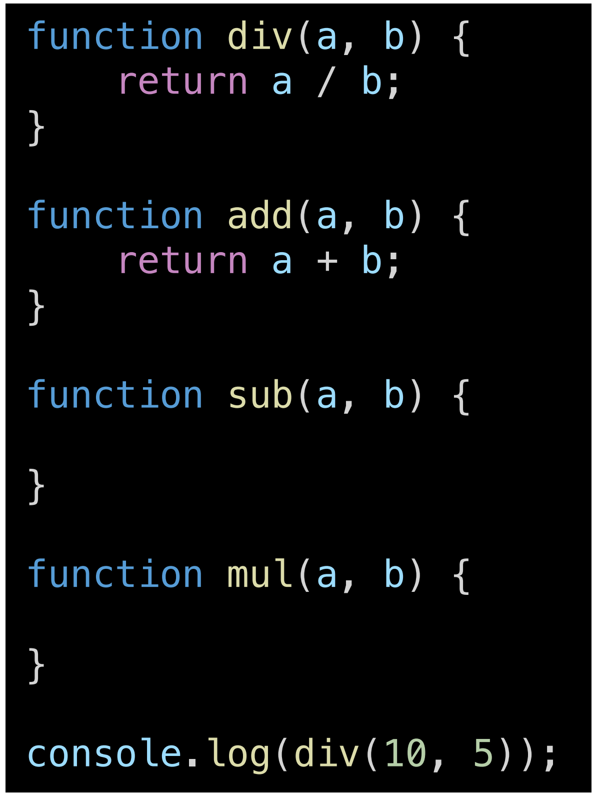
Let’s Practice – 7 (rebase)

1. Let’s make a new branch, *dev*: `git branch dev`
2. Go to *dev*: `git switch dev` (or checkout)
3. Update main.js as the code on the right and commit with a message “Added function signatures”

```
function div(a, b) {
    return a / b;
}

function add(a, b) {
    return a + b;
}

function sub(a, b) {

}

function mul(a, b) {

}

console.log(div(10, 5));
```

Let's Practice – 7 (rebase)

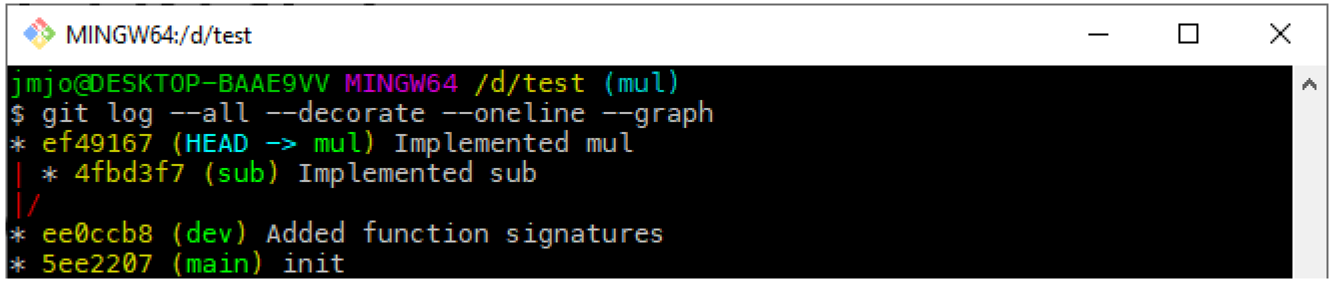
1. Let's diverge. Make two branches *sub* and *mul*: `git branch sub && git branch mul`
2. Go to *sub*, implement the *sub* function, and commit.
3. Go to *mul*, implement the *mul* function, and commit.



```
function div(a, b) {
    return a / b;
}

function add(a, b) {
    return a + b;
}

function sub(a, b) {

}

function mul(a, b) {

}

console.log(div(10, 5));
```

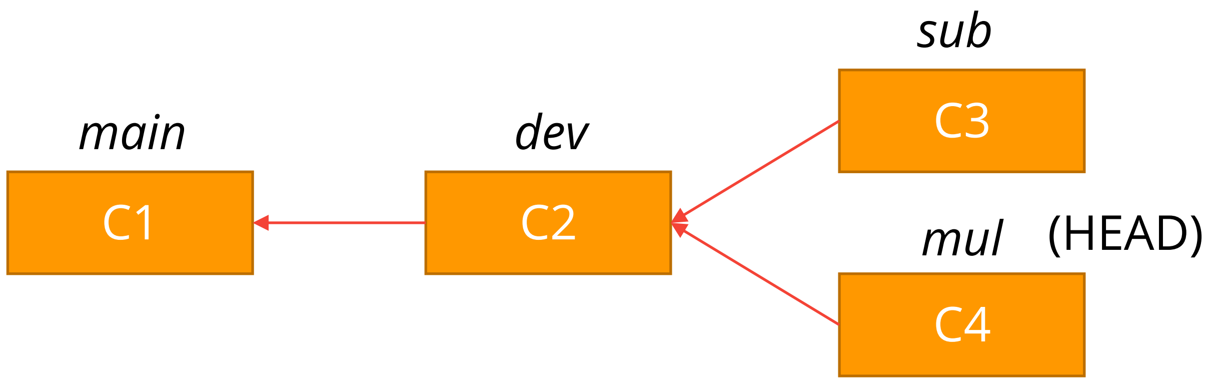
Let’s Practice – 7 (rebase)

- `git log --all --decorate --oneline --graph`



MINGW64:/d/test

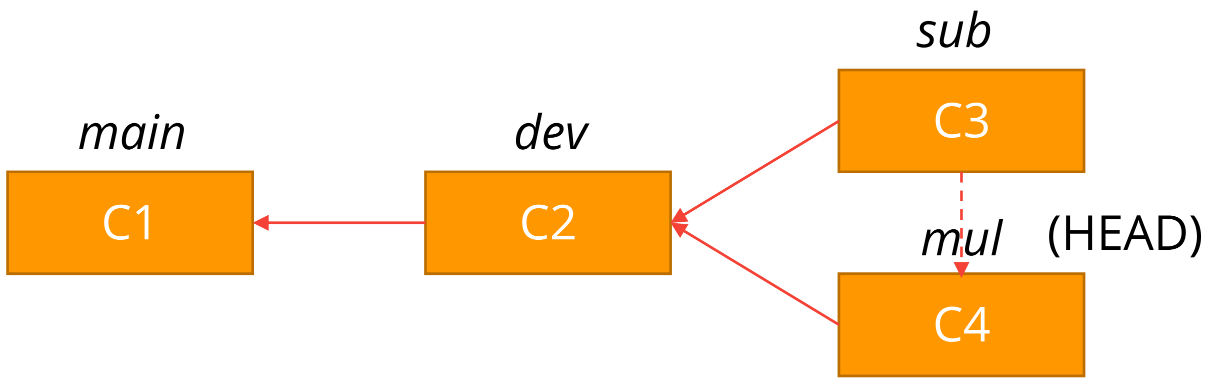
```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (mul)
$ git log --all --decorate --oneline --graph
* ef49167 (HEAD -> mul) Implemented mul
| * 4fbd3f7 (sub) Implemented sub
|/
* ee0ccb8 (dev) Added function signatures
* 5ee2207 (main) init
```



Let's Practice – 7 (rebase)

- Let's rebase C3.

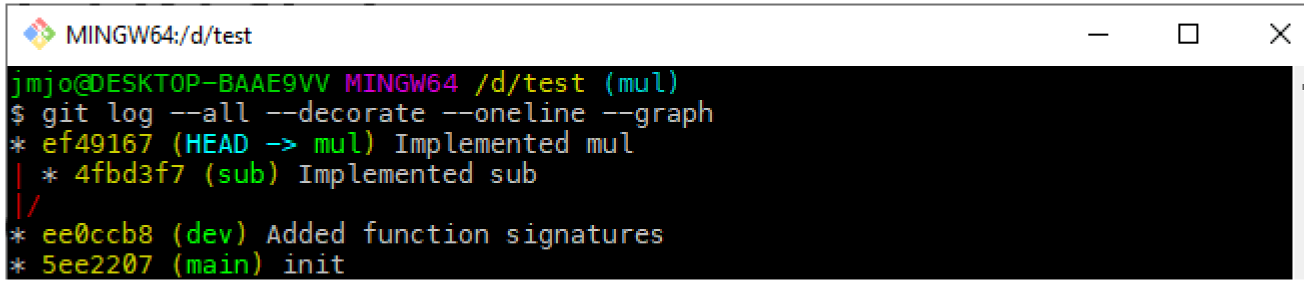
```
1. git switch sub
2. git rebase mul
3. git log --all --decorate --oneline --graph
```



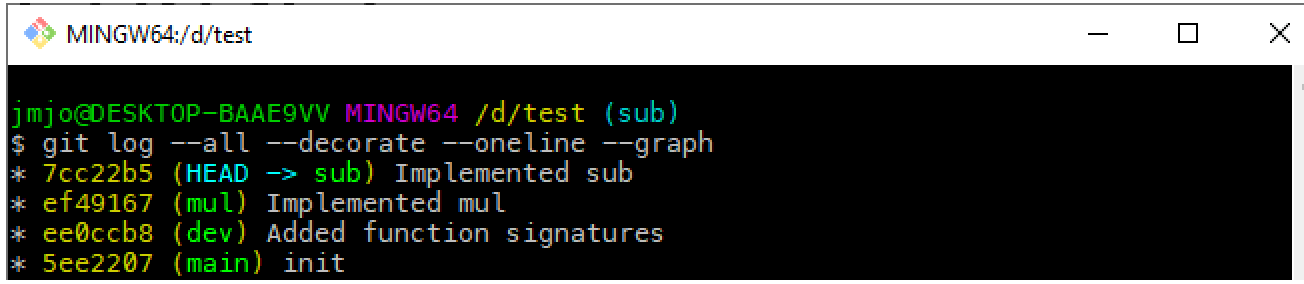
Let’s Practice – 7 (rebase)

- Two changes:
  - No branches
  - The “sub” commit appears first

- This means that the order of commits in *git log* can be different from the order where the commits were actually made.


MINGW64:/d/test

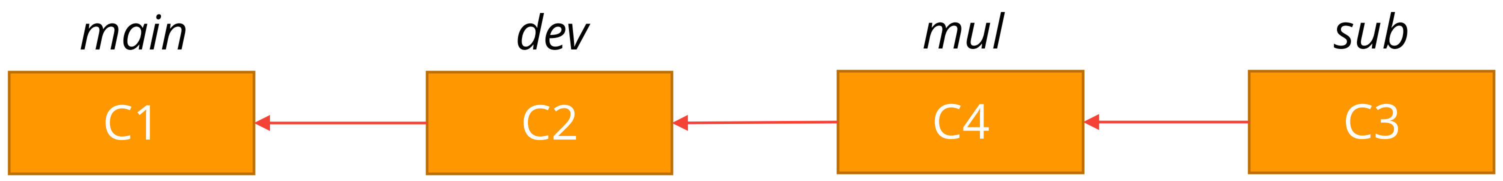
```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (mul)
$ git log --all --decorate --oneline --graph
* ef49167 (HEAD -> mul) Implemented mul
| * 4fbd3f7 (sub) Implemented sub
|/
* ee0ccb8 (dev) Added function signatures
* 5ee2207 (main) init
```


MINGW64:/d/test

```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (sub)
$ git log --all --decorate --oneline --graph
* 7cc22b5 (HEAD -> sub) Implemented sub
* ef49167 (mul) Implemented mul
* ee0ccb8 (dev) Added function signatures
* 5ee2207 (main) init
```

Let’s Practice – 7 (rebase)

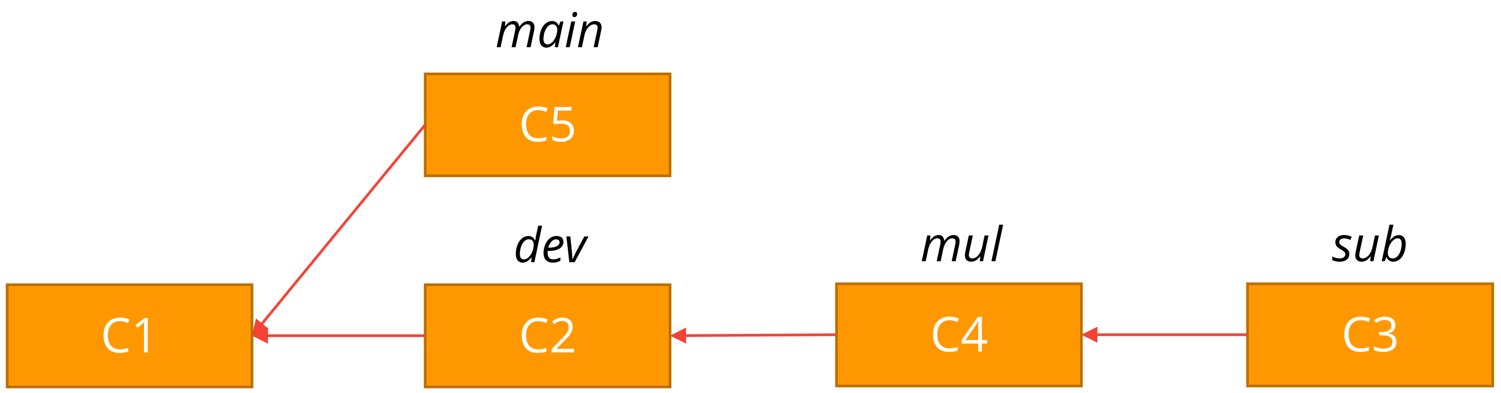
- **Author date**: the time that a commit was actually made.
  - The commit number in the graph below
- **Commit date**: the last time that a commit was modified.
  - Can be changed by rebasing
  - The (topological) order of commits in the graph



Let’s Practice – 7 (rebase)

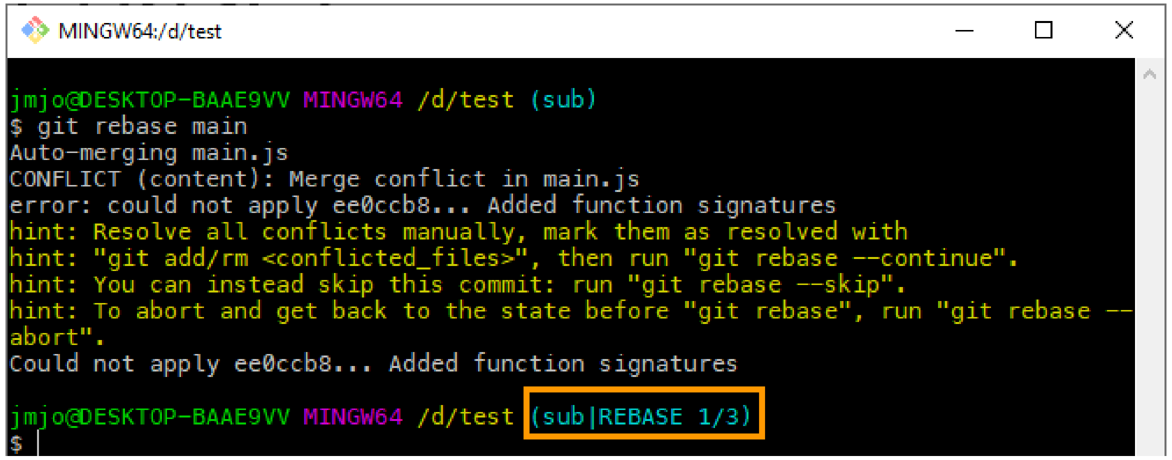
- Let’s make a rebase conflict.
- Go to *main* and add a function *mod* below the *add* function.
- This makes a conflict because Git does not know which function (*mod* vs. *sub*) should appear.

```
function div(a, b) {
    return a / b;
}

function add(a, b) {
    return a + b;
}

function mod(a, b) {
    return a % b;
}

console.log(div(10, 5));
```

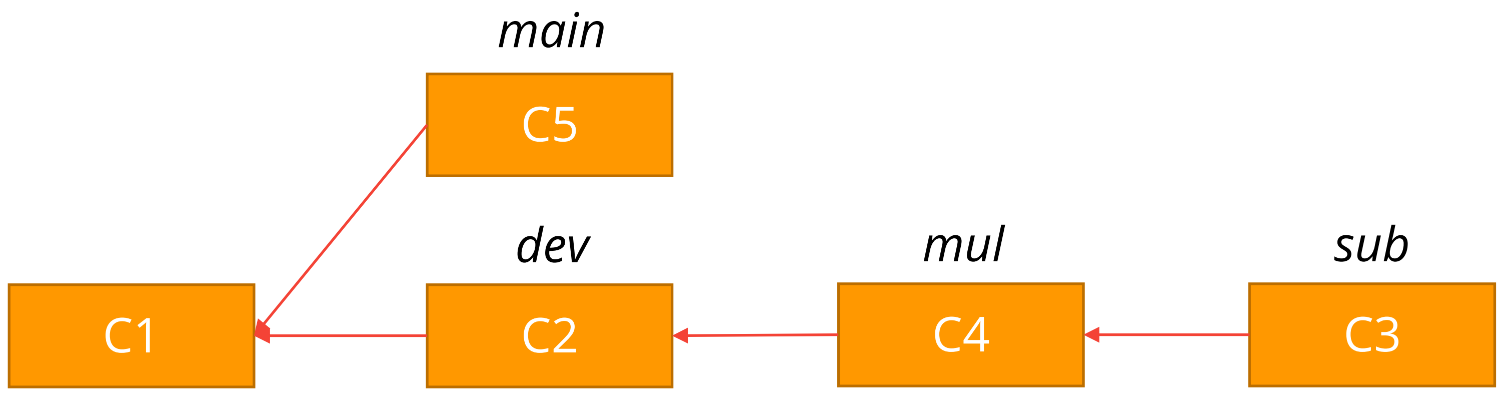


Let’s Practice – 7 (rebase)

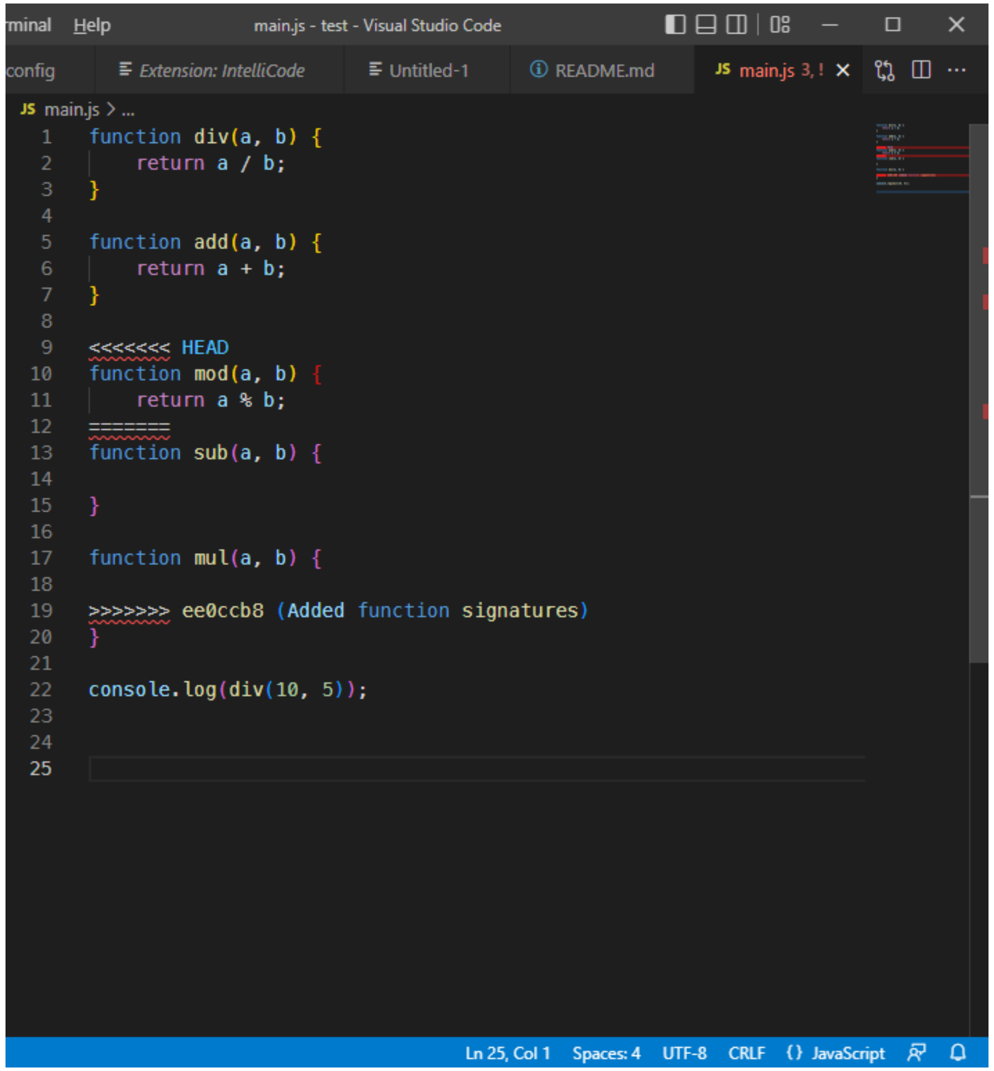
- Let’s rebase.
- `git switch sub`
- `git rebase main`
- Merge conflict happens.
- Open *main.js* and resolve the conflict.



```
MINGW64:/d/test

jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (sub)
$ git rebase main
Auto-merging main.js
CONFLICT (content): Merge conflict in main.js
error: could not apply ee0ccb8... Added function signatures
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --
abort".
Could not apply ee0ccb8... Added function signatures

jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (sub|REBASE 1/3)
$
```

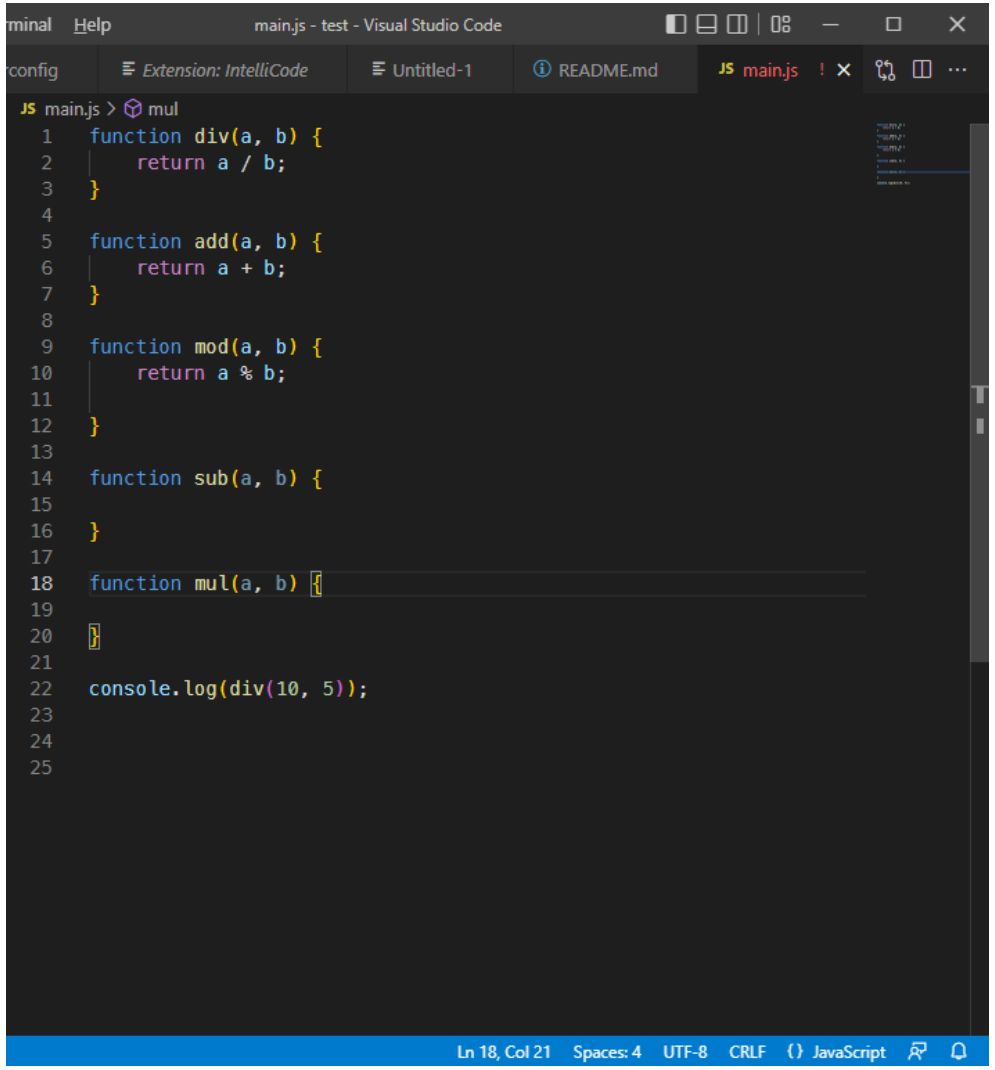


Let’s Practice – 7 (rebase)



minal Help main.js - test - Visual Studio Code
config Extension: IntelliCode Untitled-1 README.md main.js 3, !
main.js > ...
Ln 25, Col 1 Spaces: 4 UTF-8 CRLF JavaScript

```
function div(a, b) {
    return a / b;
}

function add(a, b) {
    return a + b;
}

<<<<<<< HEAD
function mod(a, b) {
    return a % b;
=======
function sub(a, b) {

}

function mul(a, b) {

>>>>>>> ee0ccb8 (Added function signatures)
}

console.log(div(10, 5));



```

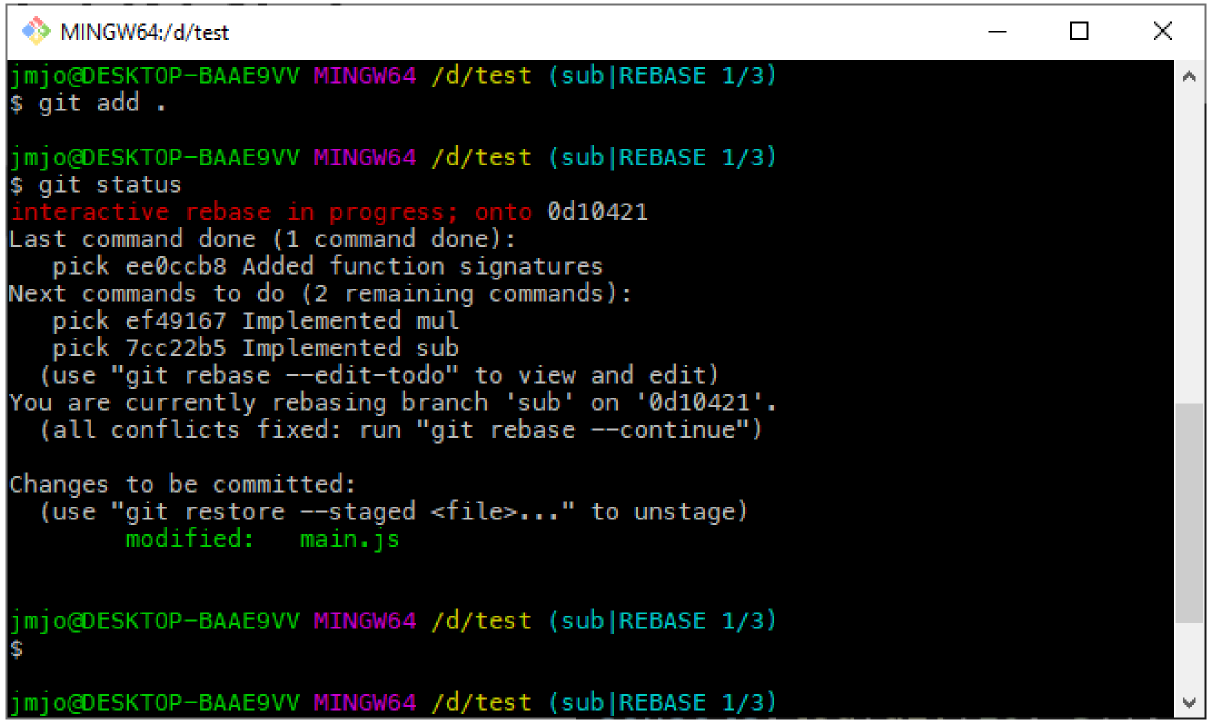


minal Help main.js - test - Visual Studio Code
config Extension: IntelliCode Untitled-1 README.md main.js !
main.js > mul
Ln 18, Col 21 Spaces: 4 UTF-8 CRLF JavaScript

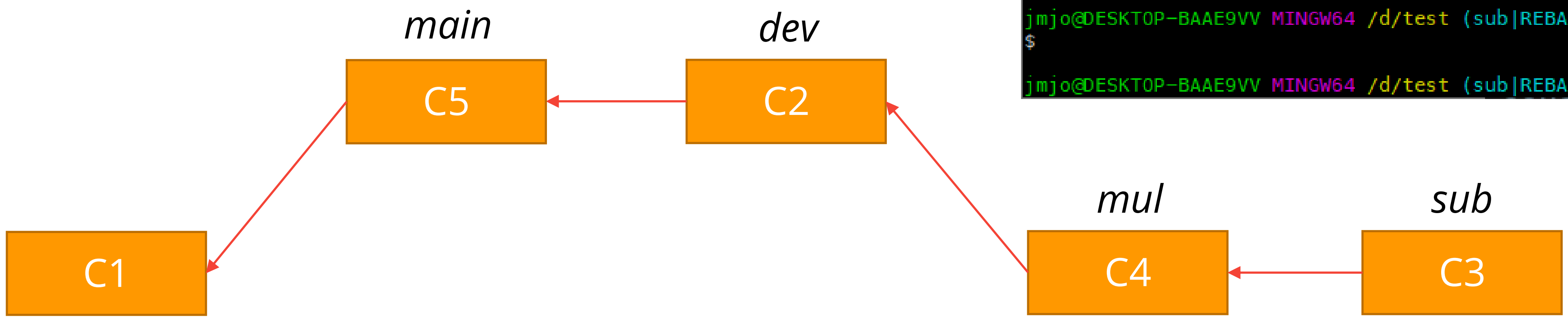
```
function div(a, b) {
    return a / b;
}

function add(a, b) {
    return a + b;
}

function mod(a, b) {
    return a % b;

}

function sub(a, b) {

}

function mul(a, b) {

}

console.log(div(10, 5));



```

Let’s Practice – 7 (rebase)

- After resolving *main.js*, add it: `git add main.js`.
- You are still rebasing! Continue it: `git rebase --continue`

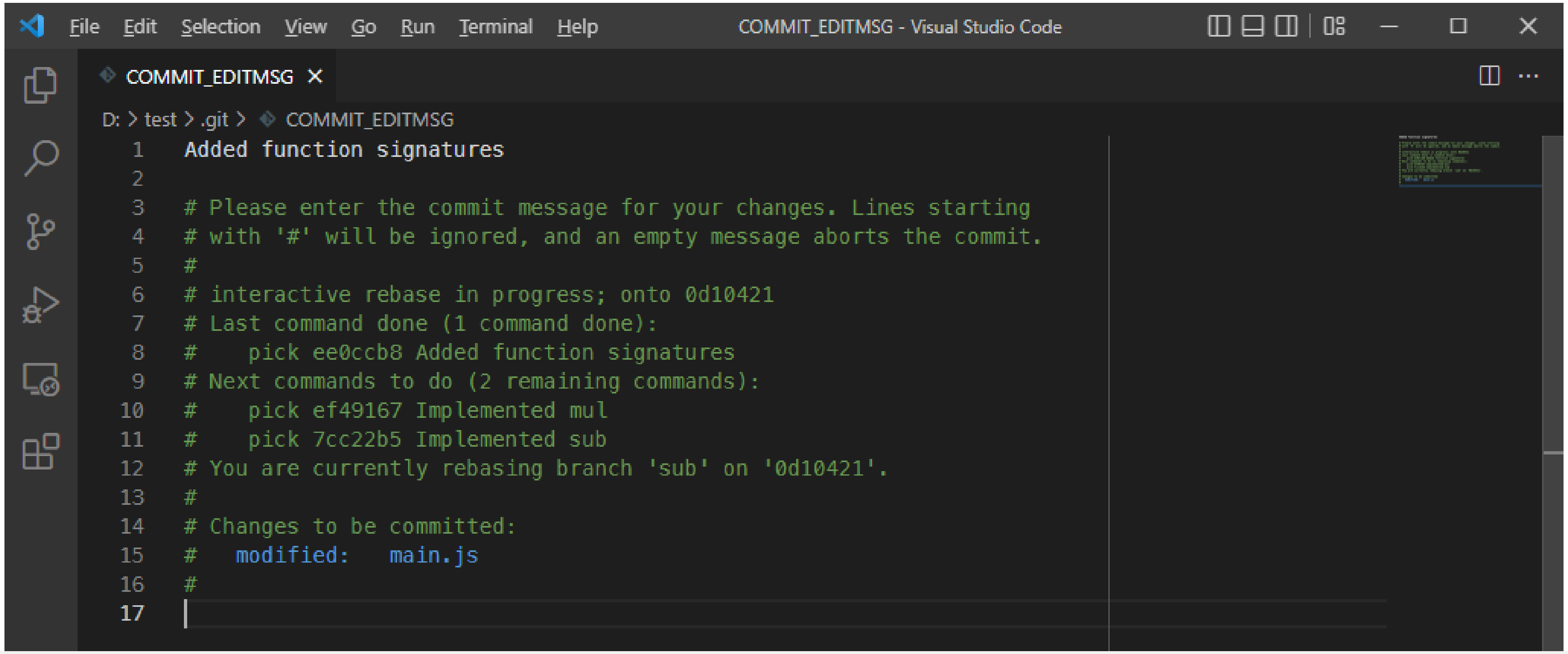


```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (sub|REBASE 1/3)
$ git add .

jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (sub|REBASE 1/3)
$ git status
interactive rebase in progress; onto 0d10421
Last command done (1 command done):
   pick ee0ccb8 Added function signatures
Next commands to do (2 remaining commands):
   pick ef49167 Implemented mul
   pick 7cc22b5 Implemented sub
  (use "git rebase --edit-todo" to view and edit)
You are currently rebasing branch 'sub' on '0d10421'.
  (all conflicts fixed: run "git rebase --continue")

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.js


jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (sub|REBASE 1/3)
$

jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (sub|REBASE 1/3)
```



Let’s Practice – 7 (rebase)

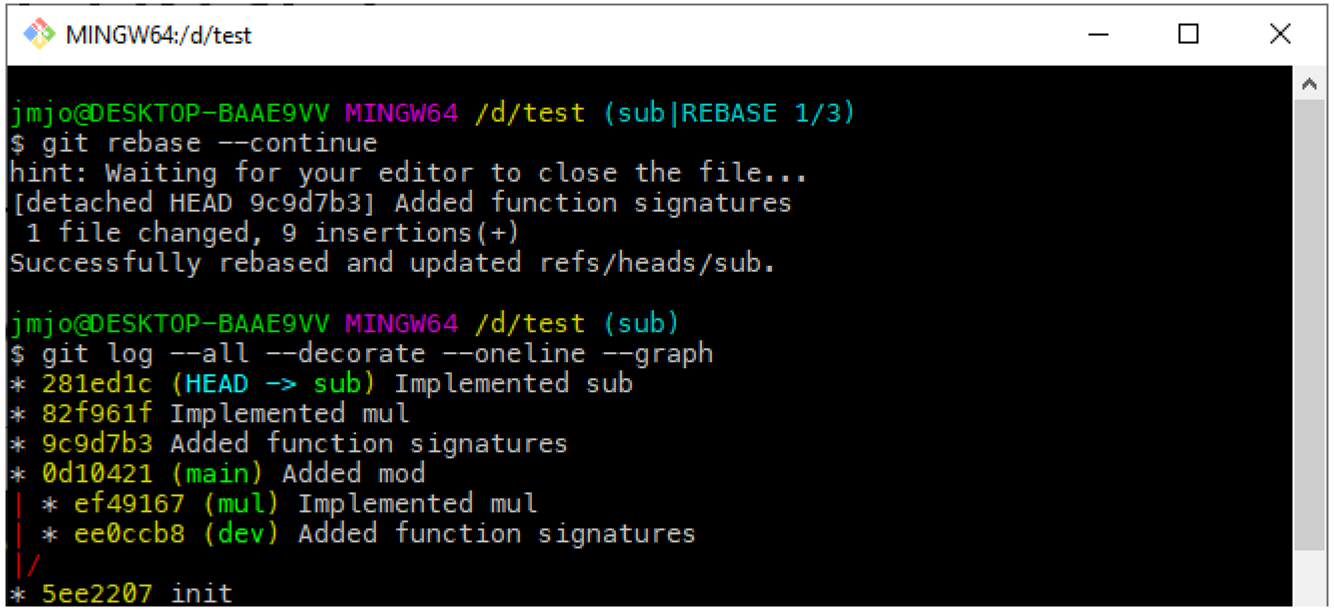
- You can change the commit message of C2.
- We will not change the message. Just close the editor.


File Edit Selection View Go Run Terminal Help
COMMIT_EDITMSG - Visual Studio Code
COMMIT_EDITMSG
D: > test > .git > COMMIT_EDITMSG

```
Added function signatures

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# interactive rebase in progress; onto 0d10421
# Last command done (1 command done):
#    pick ee0ccb8 Added function signatures
# Next commands to do (2 remaining commands):
#    pick ef49167 Implemented mul
#    pick 7cc22b5 Implemented sub
# You are currently rebasing branch 'sub' on '0d10421'.
#
# Changes to be committed:
#   modified:   main.js
#

```

Let’s Practice – 7 (rebase)

- C4 and C3 will be rebased without conflict.
- `git log --all --decorate --oneline –graph`
- Rebase created new commits from the original ones. Since every commit is rebased, you can delete old branches.

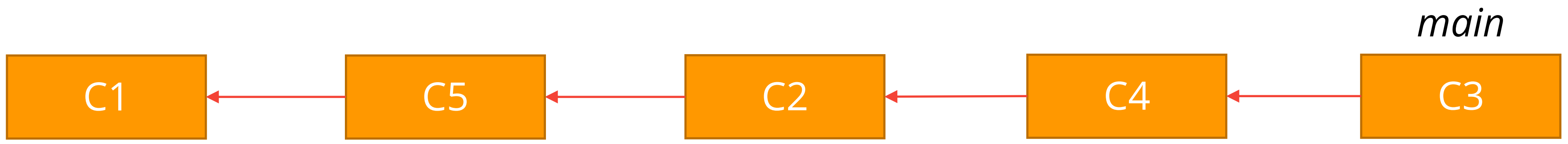


MINGW64:/d/test

```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (sub|REBASE 1/3)
$ git rebase --continue
hint: Waiting for your editor to close the file...
[detached HEAD 9c9d7b3] Added function signatures
 1 file changed, 9 insertions(+)
Successfully rebased and updated refs/heads/sub.

jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (sub)
$ git log --all --decorate --oneline --graph
* 281ed1c (HEAD -> sub) Implemented sub
* 82f961f Implemented mul
* 9c9d7b3 Added function signatures
* 0d10421 (main) Added mod
| * ef49167 (mul) Implemented mul
| * ee0ccb8 (dev) Added function signatures
|/
* 5ee2207 init
```

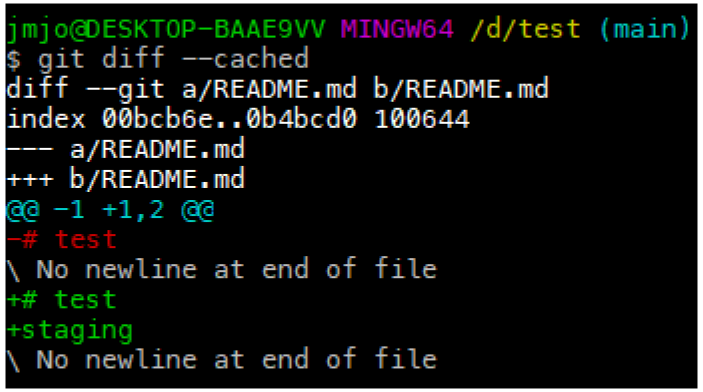
Let’s Practice – 7 (rebase)

- `git switch main`
- `git merge sub`
- `git branch –D dev`
- `git branch –D mul`
- `git branch –D sub`
- `git log --all --decorate --oneline –graph`

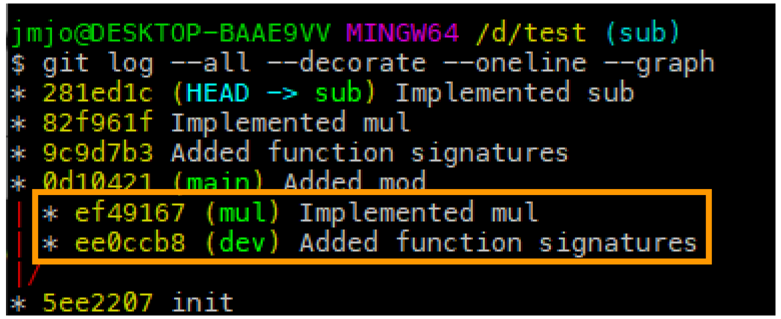


Further Questions

- Sometimes, *git diff* prints out the lines that are seemingly the same. Why?
- Where do the commits on the *mul* and *dev* branches (highlighted in orange below) go if we remove the branches?



```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (main)
$ git diff --cached
diff --git a/README.md b/README.md
index 00bcb6e..0b4bcd0 100644
--- a/README.md
+++ b/README.md
@@ -1 +1,2 @@
-# test
\ No newline at end of file
+# test
+staging
\ No newline at end of file
```



```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test (sub)
$ git log --all --decorate --oneline --graph
* 281ed1c (HEAD -> sub) Implemented sub
* 82f961f Implemented mul
* 9c9d7b3 Added function signatures
* 0d10421 (main) Added mod
| * ef49167 (mul) Implemented mul
| * ee0ccb8 (dev) Added function signatures
|/
* 5ee2207 init
```